Claims:

1. A system for reliably transmitting a request over a computer network from a client application to a server application, comprising:

a client host computer having a client protocol stack for receiving the request from a client application, wrapping the request into a request message, and transmitting the request message into the network; and

a server host computer having a server protocol stack for receiving the request message from the network, extracting the request from the request message, and delivering the request to the server application,

wherein the client application and the server application respectively provide the client protocol stack and the server protocol stack with functions including:

a function that the client protocol stack may call to obtain parameter values that govern the reliable transport of the request message to the server protocol stack for delivery to a server application; and

a function that the server protocol stack may call to obtain parameter values that govern the reliable transport of a reply wrapped in a reply message from the server protocol stack to the client protocol stack for delivery to a client application.

2. The system of claim 1, wherein the functions include:

a function for determining a retransmission timer interval for an i-th transmission of the request message and a total allowed number of transmissions of the request message to be made before an error is reported; and

a function for determining a reply cache timer for a reply cache timer.

3. The system of claim 2, wherein the functions include a function for wrapping the request in the request message and a function for wrapping the reply in the reply message.

4. The system of claim 3, wherein:

the function for wrapping the request in the request message adds a header to the request message that includes an identifying sequence number assigned by the client protocol stack and a type code identifying the type of the request message as a request; and

the function for wrapping the reply in the reply message adds a header to the reply message that includes the identifying sequence number assigned by the client protocol stack to the corresponding

request and a type code identifying the type of the reply message as a reply

5. The system of claim 4, wherein the functions include a function that returns a starting sequence number and range of allowable sequence numbers for request messages.

6. The system of claim 5, wherein the functions include a function that parses a message and returns the type and sequence number of that message, and, if the message includes a payload, the payload.

7. The system of claim 6, wherein the client protocol stack retransmits the request message if a message received from the server protocol stack contains the sequence number of the request message and a type code that indicates that it is a reply is not received before the retransmission timer interval for i-th transmission has elapsed since the request message was transmitted for the i-th time and the number of times that the request message has been transmitted is less than the total allowed number of transmissions.

8. The system of claim 7, wherein, if a message received by the client protocol stack from the server protocol stack contains the sequence number of the request message and a type code that indicates that it is a provisional reply message to the request message, then the current retransmission timer interval is modified to delay its expiration.

9. The system of claim 8, wherein, if the message type of a message received from the client protocol stack indicates that the message contains a request, delivering the request to the server application.

10. The system of claim 9, wherein, if, after delivery of the request to the server application, the server application wishes to send a provisional reply, then sending a message to the client protocol stack containing the sequence number and a message type code indicating that the message is a provisional reply to the request message.

11. The system of claim 10, wherein:

when a reply message is sent by the server protocol stack to the client protocol stack, starting the reply cache timer interval, caching the reply message in a reply cache, and destroying the cached reply message upon the expiration of the reply cache timer interval;

if the message type of a message received by the server protocol stack from the client protocol stack indicates that the message contains a request and a cached reply message has the same sequence

number, then resending the cached reply message to the client protocol stack;

if the message type of a message received by the server protocol stack from the client protocol stack indicates that the message is an acknowledgement to a cached reply message, then retaining only the sequence number in the cached reply message in the reply cache; and

when the reply cache timer interval expires, deleting the cached reply message.

12. A system for reliably transmitting a request wrapped in a request message over a computer network to a server protocol stack in a server host computer, comprising a client host computer having a client protocol stack for receiving the request from a client application, wrapping the request into a request message, and providing reliable transport of the request message to the server protocol stack, the client protocol stack when so doing calling a set of functions external to the client protocol stack and that are provided by the client application, the set of functions providing reliable transport-related services to the client protocol stack that may pre-selected to meet the requirements of the client application.

13. The system of claim 12, wherein the set of functions provided to the client protocol stack includes a function for determining a retransmission timer interval for an i-th retransmission of the request message and a total allowed number of transmissions of the request message to be made before an error is reported.

14. The system of claim 12 or claim 13, wherein the set of functions provided to the client protocol stack includes a function that the client protocol stack may call to wrap the request in the request message.

15. The system of claim 14, wherein the function that the client protocol stack may call to wrap the request in the request message adds a header to the request message including an identifying sequence number assigned by the client protocol stack and a type code identifying the type of the request message as a request.

16. The system of claim 15, wherein the set of functions provided to the client protocol stack includes a function that the client protocol stack may call to obtain a starting sequence number and range of allowable sequence numbers for request messages.

17. The system of claim 15 or claim 16, wherein the set of functions provided to the client protocol stack includes a function that the client protocol stack may call to parse a message received

from the server protocol stack and obtain the type and sequence number of that message.

18. The system of any one of claims 15 to 17, wherein the client protocol stack retransmits the request message if a reply message is received from the server protocol stack that contains the sequence number of the request message is not received before the retransmission timer interval for i-th transmission has elapsed since the request message was transmitted for the i-th time, and the number of times that the request message has been transmitted is less than the total allowed number of transmissions, but if a provisional reply message is received from the server protocol stack that contains the sequence number of the request message, then expiration of the current retransmission timer interval is delayed.

19. The system of claim 18, wherein the server host computer has a server protocol stack for receiving a request message transmitted from the client host computer, extracting a request wrapped in the request message, providing the request to a server application running on the server host computer, wrapping a reply received from the server application into a reply message, and transmitting the reply message back to the client host computer, the server protocol stack when so doing calling a set of functions external to the server protocol stack and provided by the server application, the set of functions providing reliable transport-related services to the server protocol stack than may pre-selected to meet the requirements of the server application.

20. The system of claim 19, wherein the set of functions provided to the server protocol stack includes a function for determining a reply cache interval for a reply cache timer.

21. The system of claim 20, wherein the set of functions provided to the server protocol stack includes a function that the server protocol stack may call to wrap the reply into the reply message.

22. The system of claim 21, wherein the function that the server protocol stack may call to wrap the reply in the reply message adds a header to the reply message including the identifying sequence number assigned by the client protocol stack to the corresponding request and a type code identifying the type of the reply message as a reply.

23. The system of claim 22, wherein, if the message type of a message received from the client protocol stack indicates that the message contains a request, then the request is delivered to the server application.

24. The system of claim 23, wherein, if after delivery of the request to the server application the server application wishes to send a provisional reply, then a message is sent to the client protocol stack containing the sequence number and a message type code indicating that the message is a provisional reply to the request message.

25. The system of claim 24, wherein, when a reply message is sent to the client protocol stack, then the reply cache timer interval is started, the reply message is cached in a reply cache, and the cached reply message destroyed upon the expiration of the reply cache timer interval.

26. The system of claim 25, wherein, if the message type of a message received from the client protocol stack indicates that the message contains a request and a cached reply message has the same sequence number, then the cached reply message is resent to the client protocol stack;

27. The system of claim 26, wherein, if the message type of a message received from the client protocol stack indicates that the message is an acknowledgement to a cached reply message, then only the sequence number in the cached reply message is retained in the reply cache; and

28. The system of claim 27, wherein when the reply cache timer interval expires, then the cached reply message is deleted.

29. A server host computer for use in providing reliable transport over a computer network, the server host computer having a server protocol stack for receiving a request message transmitted from a client host computer, providing a request wrapped in the request message to a server application running on the server host computer, wrapping a reply received from the server application in a reply message, and transmitting the reply message back to the client host computer, the server protocol stack when so doing calling a set of functions external to the server protocol stack and provided by the server application, the set of functions providing reliable transport-related services to the server protocol stack than may pre-selected to meet the requirements of the server application.

30. The server host computer of claim 29, wherein the set of functions includes a function for determining a reply cache interval for a reply cache timer.

31. The server host computer of claim 30, wherein the set of functions includes a function that the server protocol stack may call to wrap the reply into the reply message.

32. The server host computer of claim 31, wherein the function that the server protocol stack

may call to wrap the reply in the reply message adds a header to the reply message including the identifying sequence number assigned by the client protocol stack to the corresponding request and a type code identifying the type of the reply message as a reply.

33. The server host computer of claim 32, wherein, if the message type of a message received from the client protocol stack indicates that the message contains a request, then the request is delivered to the server application.

34. The server host computer of claim 33, wherein, if after delivery of the request to the server application the server application wishes to send a provisional reply, then a message is sent to the client protocol stack containing the sequence number and a message type code indicating that the message is a provisional reply to the request message.

35. The server host computer of claim 34, wherein, when a reply message is sent to the client protocol stack, then the reply cache interval is started, the reply message is cached in a reply cache, and the cached reply message destroyed upon the expiration of the reply cache interval.

36. The server host computer of claim 35, wherein, if the message type of a message received from the client protocol stack indicates that the message contains a request and a cached reply message has the same sequence number, then the cached reply message is resent to the client protocol stack;

37. The server host computer of claim 36 that the message is an acknowledgement to a cached reply message, then only the sequence number in the cached reply message is retained in the reply cache; and

38. The server host computer of claim 37, wherein when the reply cache timer expires, then the cached reply message is deleted.

39. A method for reliably transmitting, over a computer network from a client application to a server application, a request wrapped in a request message, in which method the client application and the server application respectively provide a client protocol stack and a server protocol stack with functions including:

a function that the client protocol stack may call to obtain parameter values that govern the reliable transport of the request message to the server protocol stack for delivery to a server application; and

a function that the server protocol stack may call to obtain parameter values that govern the reliable

transport of a reply wrapped in a reply message from the server protocol stack to the client protocol stack for delivery to a client application.

40. The method of claim 39, wherein the functions include:

a function for determining a retransmission timer interval for an i-th transmission of the request message and a total allowed number of transmissions of the request message to be made before an error is reported; and

a function for determining a reply cache timer for a reply cache timer.

41. The method of claim 40, wherein the functions include a function for wrapping the request in the request message and a function for wrapping the reply in the reply message.

42. The method of claim 41, wherein:

the function for wrapping the request in the request message adds a header to the request message that includes an identifying sequence number assigned by the client protocol stack and a type code identifying the type of the request message as a request; and

the function for wrapping the reply in the reply message adds a header to the reply message that includes the identifying sequence number assigned by the client protocol stack to the corresponding request and a type code identifying the type of the reply message as a reply

43. The method of claim 42, wherein the functions include a function that returns a starting sequence number and range of allowable sequence numbers for request messages.

44. The method of claim 43, wherein the functions include a function that parses a message and returns the type and sequence number of that message, and, if the message includes a payload, the payload.

45. The method of claim 44, wherein the client protocol stack retransmits the request message if a message received from the server protocol stack contains the sequence number of the request message and a type code that indicates that it is a reply is not received before the retransmission timer interval for i-th transmission has elapsed since the request message was transmitted for the i-th time and the number of times that the request message has been transmitted is less than the total allowed number of transmissions.

46. The method of claim 45, wherein, if a message received by the client protocol stack from the server protocol stack contains the sequence number of the request message and a type code that

indicates that it is a provisional reply message to the request message, then the current retransmission timer interval is modified to delay its expiration.

47. The method of claim 46, wherein, if the message type of a message received from the client protocol stack indicates that the message contains a request, delivering the request to the server application.

48. The method of claim 47, wherein, if, after delivery of the request to the server application, the server application wishes to send a provisional reply, then sending a message to the client protocol stack containing the sequence number and a message type code indicating that the message is a provisional reply to the request message.

49. The method of claim 37, wherein:

when a reply message is sent by the server protocol stack to the client protocol stack, starting the reply cache timer interval, caching the reply message in a reply cache, and destroying the cached reply message upon the expiration of the reply cache timer interval;

if the message type of a message received by the server protocol stack from the client protocol stack indicates that the message contains a request and a cached reply message has the same sequence number, then resending the cached reply message to the client protocol stack;

if the message type of a message received by the server protocol stack from the client protocol stack indicates that the message is an acknowledgement to a cached reply message, then retaining only the sequence number in the cached reply message in the reply cache; and

when the reply cache timer interval expires, deleting the cached reply message.

50. A method for reliably transmitting a request submitted by a client application to a client protocol stack over a computer network wrapped in a request message to a server protocol stack for delivery to a server application, in which method the client application provides the client protocol stack with a set of functions including at least one function that the client protocol stack may call to obtain parameter values that govern the reliable transport of the request message.

51. The method of claim 50, wherein the set of functions includes a function for determining a retransmission timer interval for an i-th transmission of the request message and a total allowed number of transmissions of the request message to be made before an error is reported.

52. The method of claim 50, wherein the set of functions includes a function that the client

protocol stack may call to wrap the request in the request message.

53. The method of claim 52, wherein the function that the client protocol stack may call to wrap the request in the request message adds a header to the request message that includes an identifying sequence number assigned by the client protocol stack and a type code identifying the type of the request message as a request.

54. The method of claim 53, wherein the set of functions includes a function for determining a retransmission timer interval for an i-th transmission of the request message and a total allowed number of transmissions of the request message to be made before an error is reported.

55. The method of claim 53, wherein the set of functions includes a function that the client protocol stack may call to obtain a starting sequence number and range of allowable sequence numbers for request messages.

56. The method of claim 55, wherein the set of functions includes a function that the client protocol stack may call to parse a message received from the server protocol stack and that returns the type and sequence number of that message to the client protocol stack.

57. The method of claim 56, wherein the client protocol stack retransmits the request message if a message received from the server protocol stack contains the sequence number of the request message and a type code that indicates that it is a reply is not received before the time-out interval for i-th transmission has elapsed since the request message was transmitted for the i-th time and the number of times that the request message has been transmitted is less than the total allowed number of transmissions, but if the type code and sequence number indicate that the message is a provisional reply message to the request message, then modifying the current time-out interval to delay its expiration.

58. The method of claim 57, wherein the set of functions includes a function for determining a time-out interval for an i-th transmission of the request message and a total allowed number of transmissions of the request message to be made before an error is reported.

59. A method for reliably transmitting over a computer network a request wrapped in a request message, in which method a server application provides a server protocol stack with a set of functions including at least one function that the server protocol stack calls to obtain parameter values that govern the reliable transport of a reply message from a server protocol stack to a client

protocol stack.

60. The method of claim 59, wherein the set of functions includes a function for determining a time-out interval for a reply cache timer.

61. The method of claim 60, wherein the set of functions includes a function that the server protocol stack calls to wrap the reply in the reply message.

62. The method of claim 61, wherein the function that the server protocol stack calls to wrap the reply in the reply message adds a header to the reply message that includes the identifying sequence number assigned by the client protocol stack to the corresponding request and a type code identifying the type of the reply message as a reply.

63. The method of claim 62 wherein the set of functions includes a function that the server protocol stack may call to parse a message received from the client protocol stack that returns the type and sequence number of that message.

64. The method of claim 63, wherein, if the message type of a message received from the client protocol stack indicates that the message contains a request, delivering the request to the server application.

65. The method of claim 64, wherein, if, after delivery of the request to the server application, the server application wishes to send a provisional reply, then sending a message to the client protocol stack containing the sequence number and a message type code indicating that the message is a provisional reply to the request message.

66. The method of claim 65, wherein, when a reply message is sent to the client protocol stack, starting the reply cache timer, caching the reply message in a reply cache, and destroying the cached reply message upon the expiration of the time-out interval.

67. The method of claim 66, wherein, if the message type of a message received from the client protocol stack indicates that the message contains a request and a cached reply message has the same sequence number, then resending the cached reply message to the client protocol stack;

68. The method of claim 67, wherein, if the message type of a message received from the client protocol stack indicates that the message is an acknowledgement to a cached reply message, then retaining only the sequence number in the cached reply message in the reply cache; and

69. The method of claim 68, wherein when the reply cache timer interval expires, deleting

the cached reply message.

70. A method for reliably transmitting over a network a request from a client application running on a client system to a server application running on a server system, the method comprising, in the client system:

(a) when a command is received from the client application to open a client socket,

receiving from the client application a set of pointers to a set of personality functions provided by the client application, which include

an open function,

a connect function,

a wrap request function,

a retransmit timer function,

a parse function,

a retransmit date update function,

an acknowledgement reply function, and

a close function, and

calling the open function to obtain an initial sequence number, a range of allowable sequence numbers, a maximum number of retransmissions, and header and trailer sizes, and to allocate resources to needed to support the client socket;

(b) when a command is received from the client application to connect the client socket to the server application, calling the connect function to open a connection to the server application;

(c) when a command is received from the client application to transmit a request to the server application,

determining the next sequence number,

calling the wrap request function to wrap the request in a request message having a header containing the sequence number and a message type code indicating that the request message contains a request,

calling the retransmit timer function to obtain a retransmit timer setting for

setting a retransmit timer that counts down from the retransmit timer setting,

starting the retransmit timer at the retransmit timer setting,

sending the request message to the transport layer for transmission to the server application, and

if the command to transmit the request asked that sequence number assigned to the message to be returned, then returning the sequence number to the client application;

(d) when a message is received from the transport layer,

calling the parse function to obtain the message type and sequence number of the message, and if the message type and sequence number indicate that the message is a provisional reply to the request, then modifying the retransmit timer to delay its expiration, and

if the message type and sequence number indicate that the message contains a reply to the request,

returning the reply to the client application if

the command to transmit the request specified that the reply be returned to the client application upon its receipt from the transport layer, or

the client application, since sending the command to transmit the request, has sent a command to return the reply upon its receipt from the transport layer,

but otherwise storing the reply;

(e) when a command is received from the client application to return the reply to the request and the reply has been received and stored,

returning the reply to the client application;

(f) when the reply is returned to the client application,

calling the acknowledgement reply function to send a message to the transport layer for transmission to the server application acknowledging the return of the reply to the client application, the message containing the

sequence number and an indication that the message is an acknowledgement to the reply message;

(g) if no message having a header containing the sequence number and a type code indicating that the message contains a reply has been received before the retransmit timer has expired, then repeatedly

calling the retransmit timer function to obtain a new retransmit timer setting for setting the retransmit timer,

setting and starting the retransmit timer, and

sending the request message to the transport layer for transmission to the server application,

until the retransmit timer has expired the maximum number of times or until a message having a header containing the sequence number and a type code indicating that the message contains a reply is received from the transport layer,

but if the retransmit timer has expired the maximum number of times and no such message has been received, then

reporting a transmission error to the client application and destroying any subsequent messages having a header containing the sequence number until the sequence number is assigned to a new request message;

(h) when a command is received from the client application to close the client socket,

calling the close function to free up any resources allocated to support the client socket;

and in the server system:

(i) when a command is received from the server application to open a server socket,

receiving from the server application a set of pointers to a set of personality functions, which include

the open function,

the parse function,

a wrap reply function,

a reply cache timer function, and

the close function, and

calling the open function to allocate resources needed to support the server socket;

(j) when a command is received from the server application to receive a request from the client application and then a message is received from the client application,

calling the parse function to obtain the message type and sequence number of the message and, if the message type indicates that the message contains a request, the request, and then returning the request to the server application process;

(k) if, after delivery of the request to the server application, a command is received from the server application to send a provisional reply, then

sending a message to the transport layer for transmission to the client application containing the sequence number and a message type code indicating that the message is a provisional reply to the request message;

(l) when a command is received from the server application to transmit a reply to the client application,

calling the wrap reply function to wrap the reply in a header containing the sequence number and a message type code indicating that the request message contains a reply,

calling the reply cache timer function to obtain a reply cache timer setting for setting a reply cache timer that counts down from the setting,

starting the reply cache timer,

sending the resulting message to the transport layer for transmission to the client application, and

caching the reply message;

(m) when a further message is received from the client application,

calling the parse function to obtain the message type and sequence number of

the message and

if the message type indicates that the message contains a request and a cached reply message has the same sequence number, then

resending the cached reply message to the transport layer for transmission to the client application, and

if a message contains an indication that the message is an acknowledgement to a cached reply message, then

deleting everything except the sequence number from the cached reply message;

(n) when the reply cache timer expires, then

deleting the cached reply message; and

(o) when a command is received from the server application to close the server socket,

calling the close function to free up any resources allocated by the personality functions.